\Orchestrating a brighter world NEC

The Slow NVM (R)evolution
in the context of
Next Generation Postprocessing

5. December 2016, Stuttgart

Workshop on Sustained Simulation Performance

Erich Focht

NEC HPC Europe

\Orchestrating a brighter world

NEC brings together and integrates technology and expertise to create
the ICT-enabled society of tomorrow.
We collaborate closely with partners and customers around the world,
orchestrating each project to ensure all its parts are fine-tuned to local needs.

Every day, our innovative solutions for society contribute to
greater safety, security, efficiency and equality,
and enable people to live brighter lives.

“The K Problem”

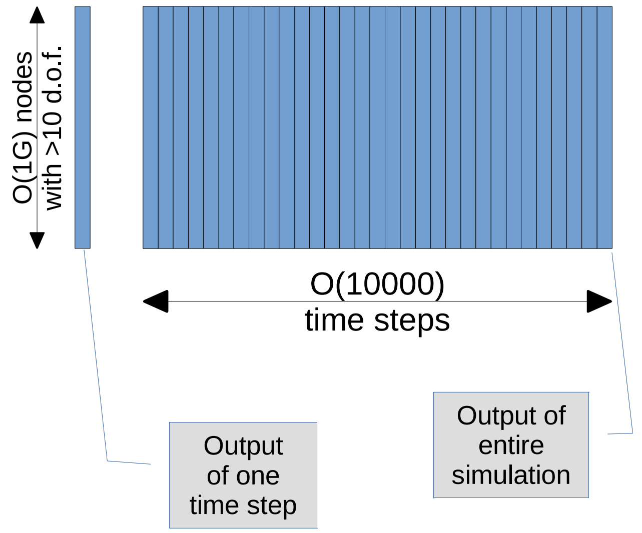
- Two things came together:
  - Uwe **K**üster
  - The **K**-Operator
- Hope for new ways of simulating complex non-linear dynamics problems
- Related or not... Uwe Küster sees upcoming strong demand for new ways of post-processing
  - *Correlate outputs in time direction!*

Next Generation Post-Processing

- Is a big data problem, much more than it was before
  - Problem size: 1 GigaNodes
  - Degrees of freedom per node: >10
  - Output per time step: > 80GB
  - With O(1000) – O(10000) time steps: O(100TB) – O(1PB) Output per job
    - Local examples (last year):
      Matthias Meinke: 100TB output
      Meteo Uni Hohenheim: 330TB output with 3km resolution
  - Write once, read multiple. Huge amount of data to be read!
  - Can't afford to store entire data forever
  - Analyse, "compress" by extracting knowledge, remove output
  - Don't forget: this is about processing ALL OUTPUT, not just one time step!

The K - Way

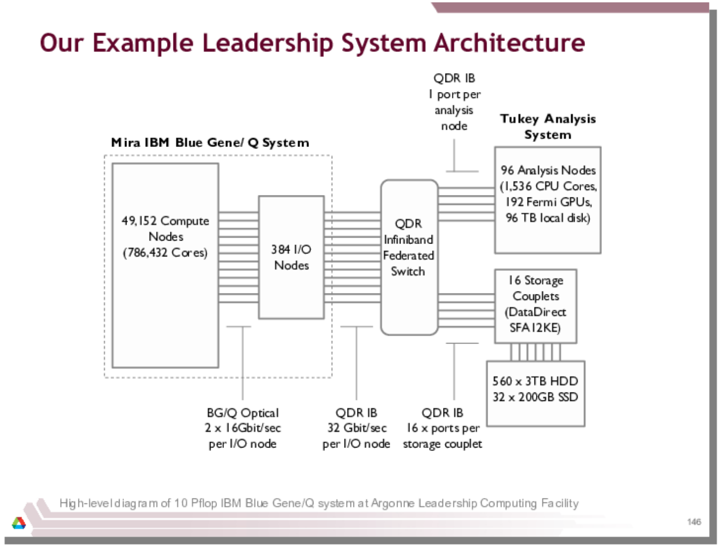
- Treat entire simulation output as a matrix
  - Find Eigenvalues
  - Spectral analysis, FFTs
  - Matrix multiply, Transpose
  - **Big data analysis with HPC methods**
- Challenges
  - Size: 100TB - 1PB of data!
  - With 100GB/s bandwidth filesystem: 1000 - 10000s to read data in once.
  - Dense matrix operations! Bandwidth!



\Orchestrating a brighter world NEC

Postprocessing: State of the Art

- Following figures taken from:
  "HPC I/O for Computational Scientists": Rob Latham et al, Course at Argonne National Lab.

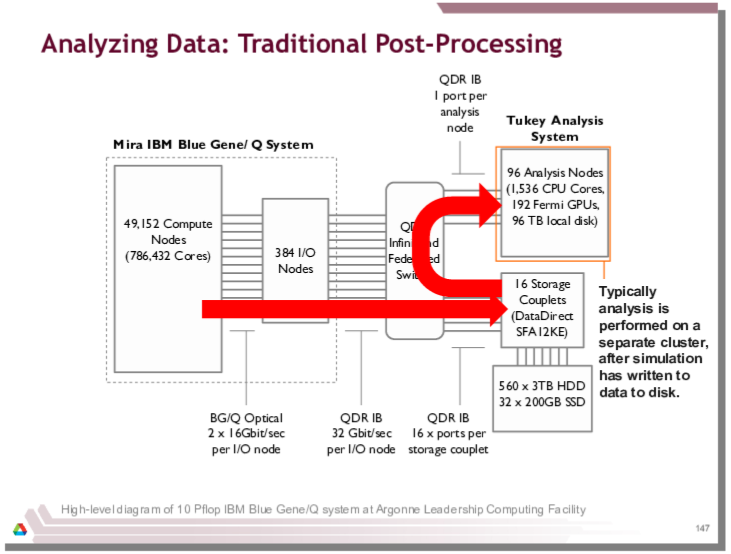


Our Example Leadership System Architecture

Mira IBM Blue Gene/ Q System
49,152 Compute Nodes (786,432 Cores)
384 I/O Nodes
QDR Infiniband Federated Switch
QDR IB 1 port per analysis node
Tukey Analysis System
96 Analysis Nodes (1,536 CPU Cores, 192 Fermi GPUs, 96 TB local disk)
16 Storage Couplets (DataDirect SFA12KE)
560 x 3TB HDD 32 x 200GB SSD
BG/Q Optical 2 x 16Gbit/sec per I/O node
QDR IB 32 Gbit/sec per I/O node
QDR IB 16 x ports per storage couplet

High-level diagram of 10 Pflop IBM Blue Gene/Q system at Argonne Leadership Computing Facility

146



Analyzing Data: Traditional Post-Processing

Mira IBM Blue Gene/ Q System
49,152 Compute Nodes (786,432 Cores)
384 I/O Nodes
QDR IB 1 port per analysis node
Tukey Analysis System
96 Analysis Nodes (1,536 CPU Cores, 192 Fermi GPUs, 96 TB local disk)
16 Storage Couplets (DataDirect SFA12KE)
560 x 3TB HDD 32 x 200GB SSD
BG/Q Optical 2 x 16Gbit/sec per I/O node
QDR IB 32 Gbit/sec per I/O node
QDR IB 16 x ports per storage couplet

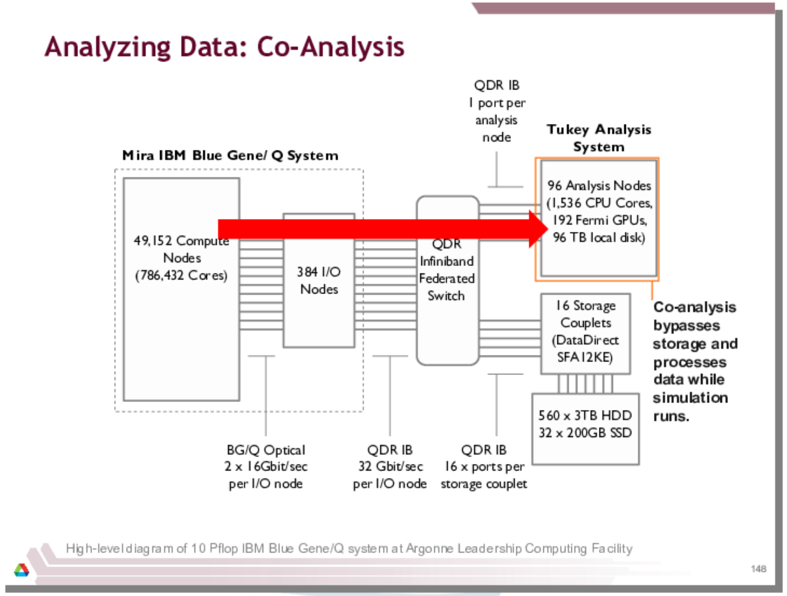
Typically analysis is performed on a separate cluster, after simulation has written to disk.

High-level diagram of 10 Pflop IBM Blue Gene/Q system at Argonne Leadership Computing Facility

147

Postprocessing: State of the Art (continued)

- Attempts to deal with huge amounts of data:



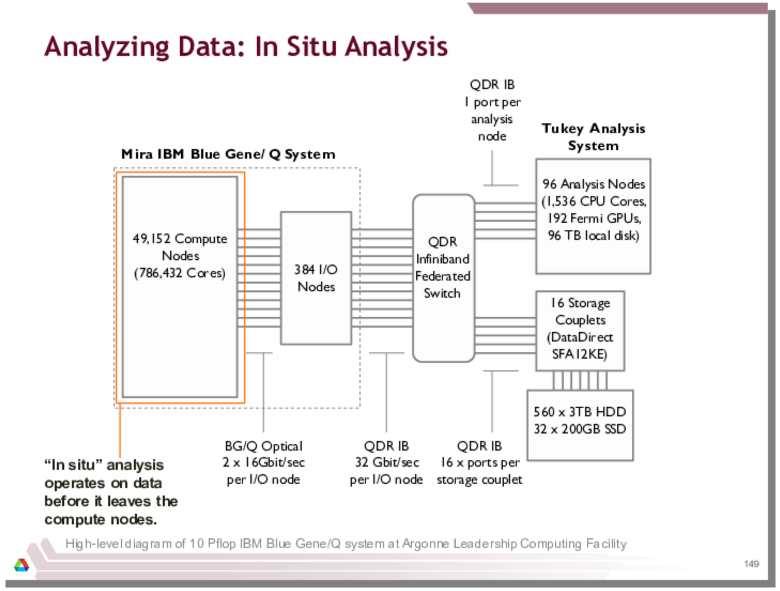
Analyzing Data: Co-Analysis

Co-analysis bypasses storage and processes data while simulation runs.

High-level diagram of 10 Pflop IBM Blue Gene/Q system at Argonne Leadership Computing Facility

148

Mira IBM Blue Gene/ Q System
49,152 Compute Nodes (786,432 Cores)
384 I/O Nodes
QDR Infiniband Federated Switch
Tukey Analysis System
96 Analysis Nodes (1,536 CPU Cores, 192 Fermi GPUs, 96 TB local disk)
16 Storage Couplets (DataDirect SFA12KE)
560 x 3TB HDD 32 x 200GB SSD
QDR IB 1 port per analysis node
BG/Q Optical 2 x 16Gbit/sec per I/O node
QDR IB 32 Gbit/sec per I/O node
QDR IB 16 x ports per storage couplet



Analyzing Data: In Situ Analysis

"In situ" analysis operates on data before it leaves the compute nodes.

High-level diagram of 10 Pflop IBM Blue Gene/Q system at Argonne Leadership Computing Facility

149

Mira IBM Blue Gene/ Q System
49,152 Compute Nodes (786,432 Cores)
384 I/O Nodes
QDR Infiniband Federated Switch
Tukey Analysis System
96 Analysis Nodes (1,536 CPU Cores, 192 Fermi GPUs, 96 TB local disk)
16 Storage Couplets (DataDirect SFA12KE)
560 x 3TB HDD 32 x 200GB SSD
QDR IB 1 port per analysis node
BG/Q Optical 2 x 16Gbit/sec per I/O node
QDR IB 32 Gbit/sec per I/O node
QDR IB 16 x ports per storage couplet

Nice, but only suited for analysis of one time step!

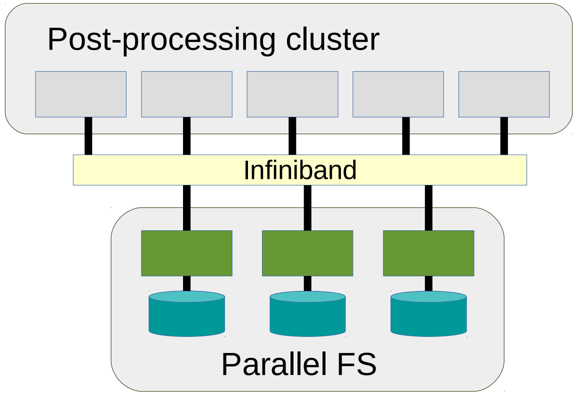
Approaches

Target Space

- Learn & start with lower end: 100TB output cases
- ~ 12 nodes for post-processing in parallel
  - ~100GB RAM / node, ~1.2TB RAM in total
- Dedicated for limited time for one user to post-process his job
- Output data is on a parallel file system, but needs to be deleted after post-processing
- POSIX is irrelevant
- Post-processing software has to be written or adapted to IO

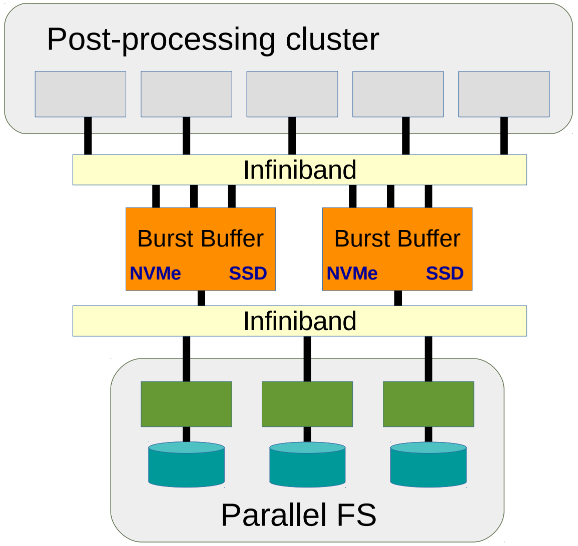
Traditional Approach

- Post-processing cluster connected to parallel file system
  - Simplest setup
  - Bandwidth to PFS is limited
    - Parallel FS
    - Post-processing nodes
  - *On post-processing nodes side:*
    12 post-processing nodes:
    12 * 3-5GB/s, total 36-60GB/s
  - Loading 100TB: min. 2000-3000s
    - Will be loaded many times!
    - Lost in IO



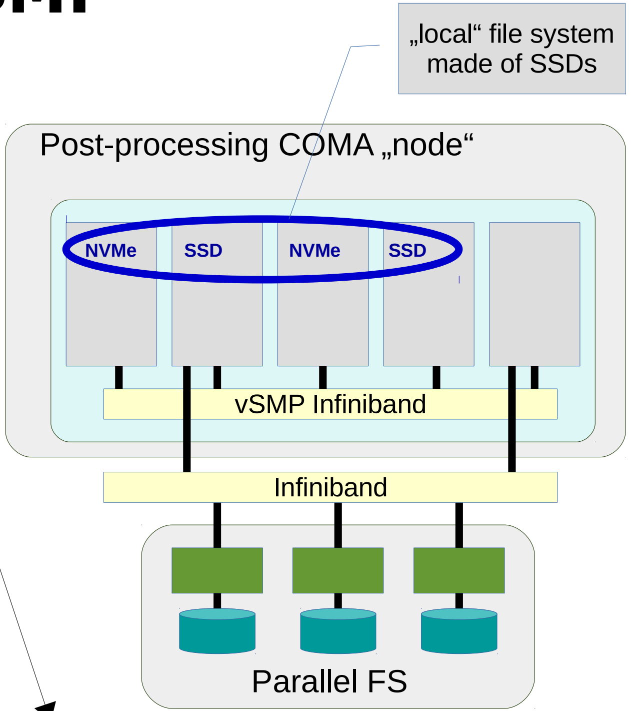
Burst Buffers and Parallel File System

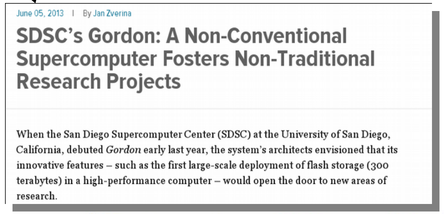
- Post-processing cluster connected to burst buffers connected to parallel file system
  - More expensive than traditional approach
  - Bandwidth to burst buffer nodes could be sufficient
    - Caching? How coherent? IME alike?
    - Another parallel FS over burst buffers?
    - Entire data set should fit into burst buffers
  - Challenging
    - Say … 5 burst buffer nodes
    - 5 * 20GB/s towards post-processing cluster
    - 5 * O(5GB/s) bandwidth BB nodes to PFS
    - > 5 * 20TB NVMe / SSDs
  - BB nodes dedicated to this post processing instance
  - Loading 100TB:
    - O(3-4000s) to BB nodes, once
    - O(1000s) from BB nodes to post-processing
  - A faster "traditional approach".



“Inverse Virtualization”: vSMP

- Create vSMP node with ScaleMP SW
  - eg. San Diego Supercomputer Center
    - GORDON
  - Aggregate post-processing nodes to one vSMP node using COMA
    - Use NVMe and SSDs on any of the nodes as if they were local
    - Still possible to access parallel FS
  - What is the impact of COMA?
  - VSMP “cache” coherency network is limiting factor
  - Nevertheless: interesting approach!



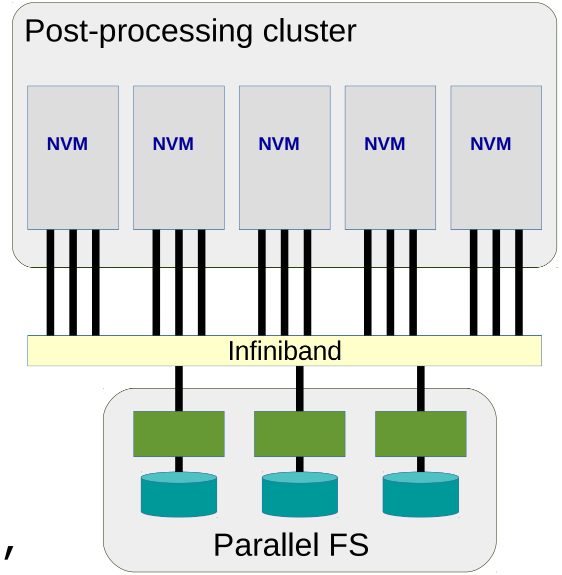

June 05, 2013 | By Jan Zverina

SDSC’s Gordon: A Non-Conventional Supercomputer Fosters Non-Traditional Research Projects

When the San Diego Supercomputer Center (SDSC) at the University of San Diego, California, debuted *Gordon* early last year, the system’s architects envisioned that its innovative features – such as the first large-scale deployment of flash storage (300 terabytes) in a high-performance computer – would open the door to new areas of research.

“No Global File System” Approach

- Post-processing nodes with NVM and fat IB connectivity
  - Actually “burst buffer” nodes
    - But software makes the difference
    - >10TB in non-volatile memory / node
    - *No parallel file system for storing output matrix*
    - 2 * EDR/OPA: ~20GB/s IB connectivity
  - MPI bandwidth between nodes: ~20GB/s, >240GB/s aggregated.
    - Usable for transfer of submatrices between nodes.
  - Use data “in place”, compute directly from NVM/PMEM!



“No Global File System” Approach

- Post-processing nodes with NVM and fat IB connectivity

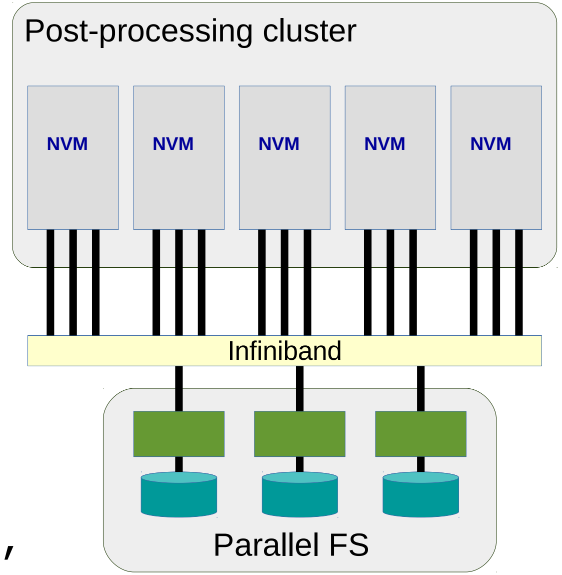
Actually…
we want at least 100TB of RAM
(or something similar)
in a small and cheap system.

„in-memory“ - „out-of-core“

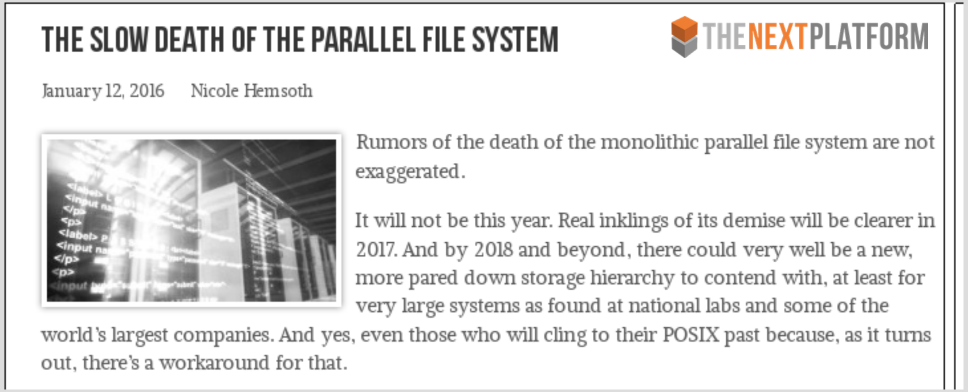
/s,

- Usable for transfer of submatrices between nodes.

– Use data “in place”, compute directly from NVM/PMEM!



“No Global File System” Approach



THE SLOW DEATH OF THE PARALLEL FILE SYSTEM

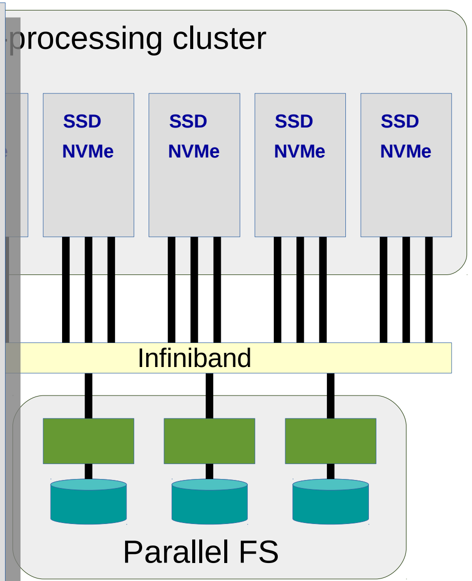
THENEXTPLATFORM

January 12, 2016 Nicole Hemsoth

Rumors of the death of the monolithic parallel file system are not exaggerated.

It will not be this year. Real inklings of its demise will be clearer in 2017. And by 2018 and beyond, there could very well be a new, more pared down storage hierarchy to contend with, at least for very large systems as found at national labs and some of the world's largest companies. And yes, even those who will cling to their POSIX past because, as it turns out, there's a workaround for that.

- POSIX semantics & consistency not needed over the entire name-space of a data center. Only needed within the name-space spanned by a job. Even there we can work around it.
- Best example: DAOS container.
  - Domain specific IO: NetCDF, HDF5, ...
- Other approaches: BatchFS, DeltaFS, MarFS...
  - Metadata only within job, preferably managed by job nodes
  - Use object storage as back end

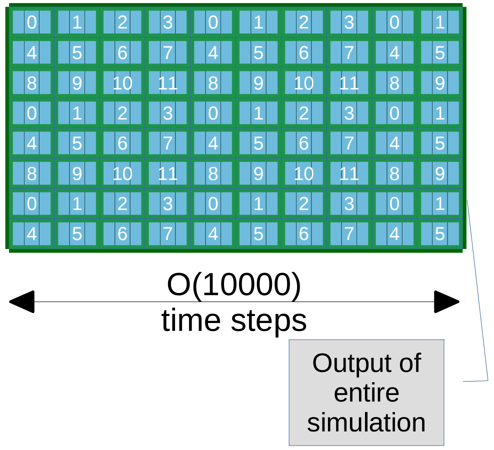


Matrix Representation

- SCALAPACK & BLACS!
  - Two-dimensional block-cyclic distribution
- Decompose output matrix into sub-matrices of same size
- Each submatrix is stored contiguously in NVM
  - mmaped
  - Byte addressable (!)
  - Persistent: flush or msync needed
- Byte address of each submatrix reconstructable with metadata block MD
  - Locate & reconstruct matrix after node failure or reboot

BLACS processor grid

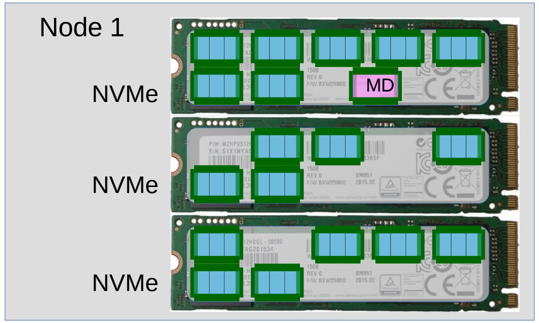
0	1	2	3
4	5	6	7
8	9	10	11



0	1	2	3	0	1	2	3	0	1
4	5	6	7	4	5	6	7	4	5
8	9	10	11	8	9	10	11	8	9
0	1	2	3	0	1	2	3	0	1
4	5	6	7	4	5	6	7	4	5
8	9	10	11	8	9	10	11	8	9
0	1	2	3	0	1	2	3	0	1
4	5	6	7	4	5	6	7	4	5

Matrix Representation

- For example:
  - Each submatrix is a file
  - Or: all submatrices owned by a node are in one file. One mmap for each submatrix.



- TODO: Replace initialization of local block submatrix usually done in PXELSET()

```
CALL DESCINIT(DESCA, M, N, MB, NB, RSRC, CSRC, CONTEXT, LLD, IERR)
...
CALL PDELSET( A, IA, JA, DESCA, ALPHA )
...
```

- TODO: import/export matrices from/to parallel FS

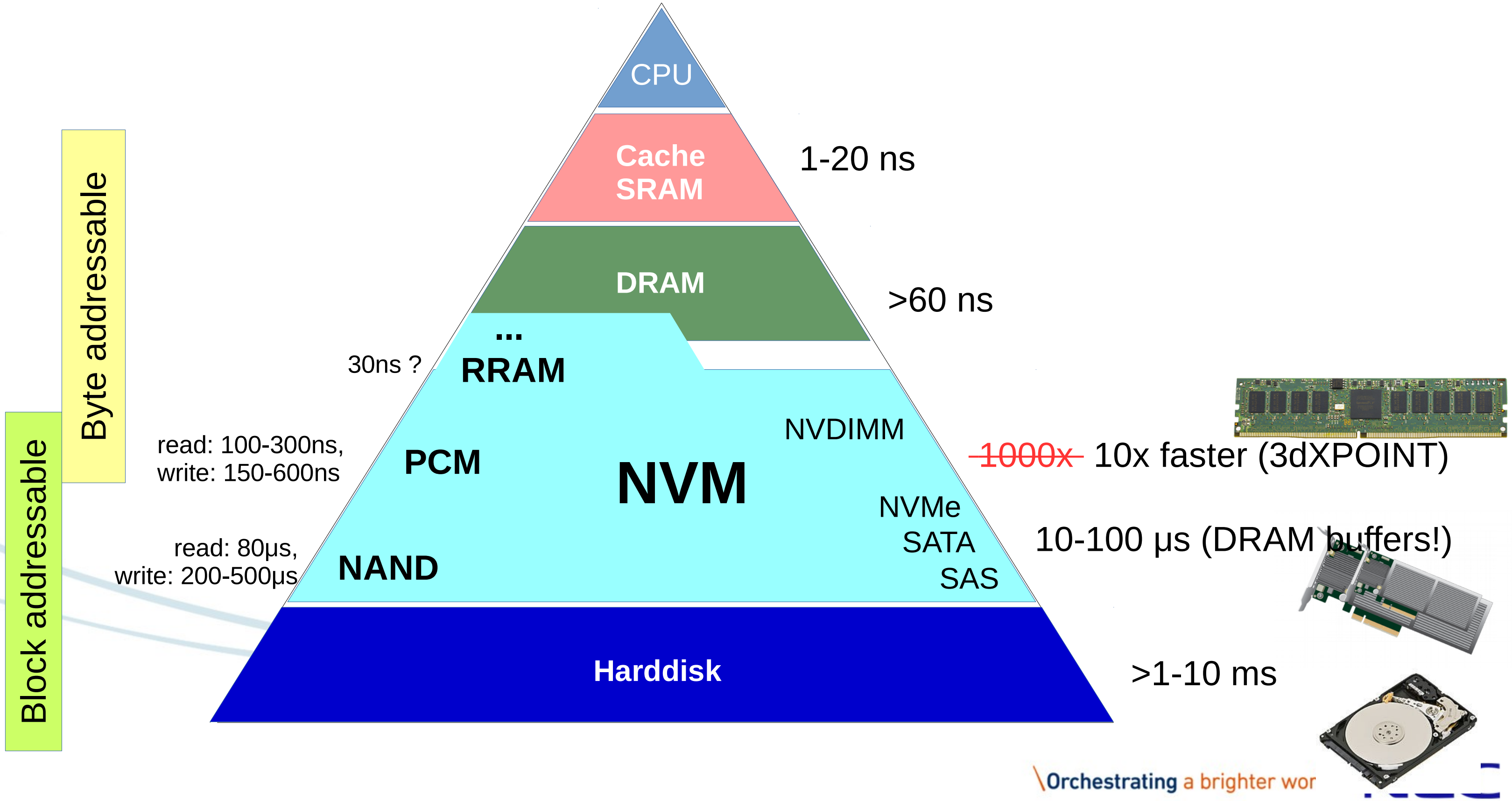
Software and API

- PMEM: Persistent Memory
  - Persistent Memory project, SNIA
  - Mapped, byte addressable
  - Mechanisms for finding own data in PMEM after reboot
- Matrices stored in distributed PMEM
  - Compute from there, transfer from there, recover from there
  - Effective virtual memory of size of PMEM
- SCALAPACK / BLACS
  - Small modifications, transfer submatrices from PMEM through RDMA
- MPI multirail Infiniband
- Lustre / Parallel FS client for loading the output (once)

Issues, Today

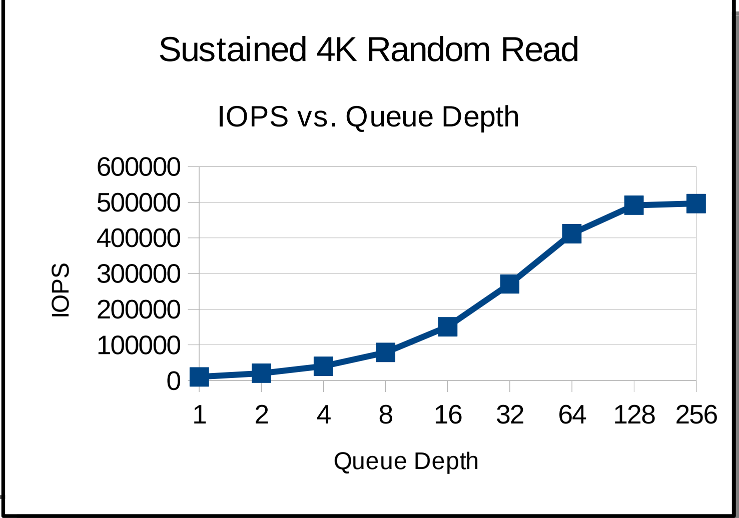
- NVM: byte addressable
  - Persystent memory PMEM ... not yet here (though big promisses were made)
  - NVMe great (also price-wise) but not exactly what we need
- Might need multiple layouts of the matrix, i.e. different distributions (eg. Transpose).
  - Increase NVM space or number of servers
- 2 x EDR bandwidth, able to use 20-24GB/s?
  - NUMA challenges, zero copy wherever possible
- Latency!

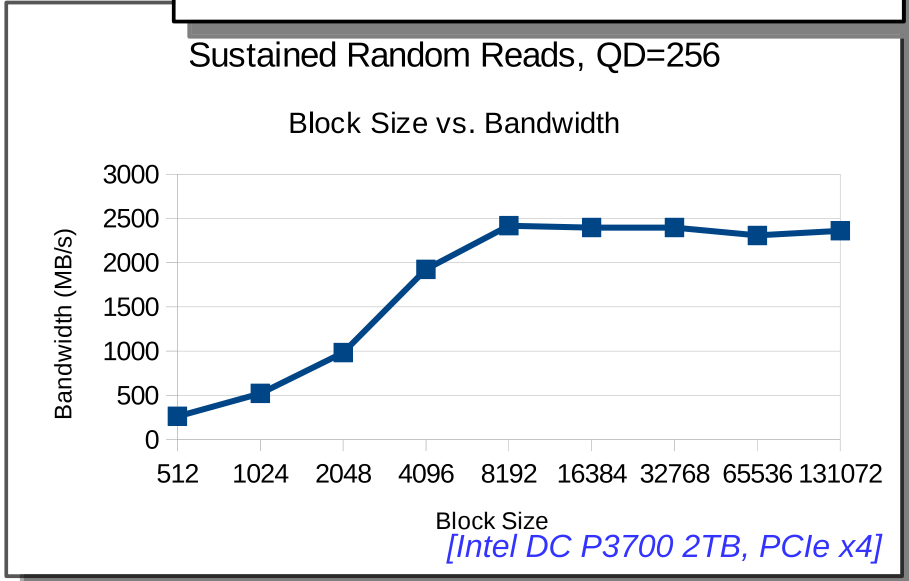
NVM in Memory Hierarchy



Available Today: NVMe

- Efficient and streamlined command set
- Huge parallelity (64k command queues)
- One IO register write for one cmd issue
- Jump in IOPS and BW!
  - 3x SAS IOPS
  - 6-8x SATA IOPS
  - 3x SAS read BW
  - 6x SATA read BW
- Gained 20µs vs. SAS/SATA due to missing SCSI SW stack
- Not byte addressable :-(

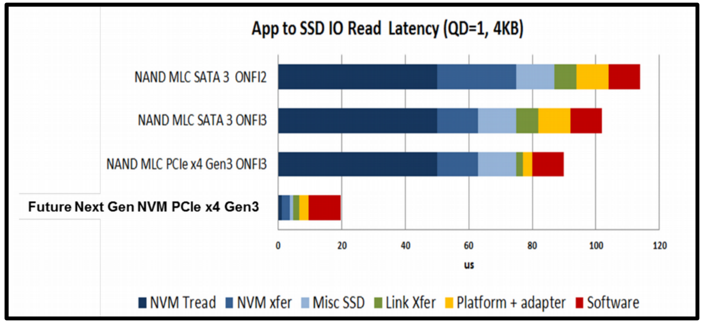




[Intel DC P3700 2TB, PCIe x4]

Scaling NVM (express)

- Various form factors
  - M.2, U.2 (x4, 2x x2 dual port)
  - PCIe x4, x8, x16 (!)
    - x8: >6GB/s read bandwidth
    - x16: 10GB/s read bandwidth, 2M IOPS



- 2015
  - Scalable at rack level through PCIe switches
    - NVMe: EMC DSSD D5: 10M IOPS, 100GB/s, 100μs latency
    - (SAS SSD: NetApp EF560: 825k IOPS, 12GB/s, 300-800μs latency)
- 2016
  - Scale at data center level
    - NVMe over Fabrics
      - < 6μs additional latency!
    - NVMe over Ethernet (Apeiron/NEC)
      - < 3μs additional latency (round-trip)

NVDIMM in 2016?

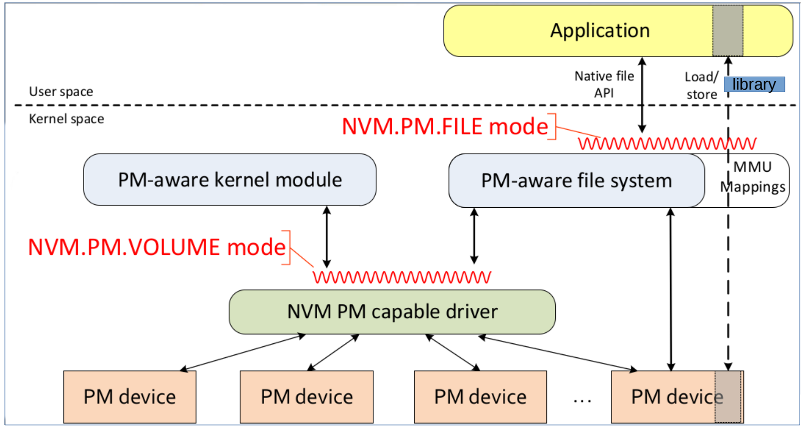
- NVDIMM-N
  - DRAM, backed by flash
  - Capacity of a DRAM DIMM (eg. 16GB)
- NVDIMM-F
  - NAND flash block device with DDR3/4 interface
- Diablo “Memory1”
  - NVDIMM-F, 256(-512)GB, software pushes pages from DRAM into it
  - Effectively extends server memory size
- New technology? PCM? MRAM?
  - 3d Xpoint... 2017? Capacity? Density? Price?

NVM Programming Model(s)

- NVMe: not really...
  - it’s a block device, sector addressable
  - ... file system or KV store
- Mnemosyne, NV Heaps (2011!)
  - Lightweight persistent memory, consistent updates, persistent regions
- SNIA PMEM.IO
  - Storage Networks Industry Association
  - Big effort to prepare for PMEM
  - Programming model for PMEM
  - ***NVM Library***: optimized for PMEM but behaves reasonably on other types of NVM like SSDs

PMEM Programming Model

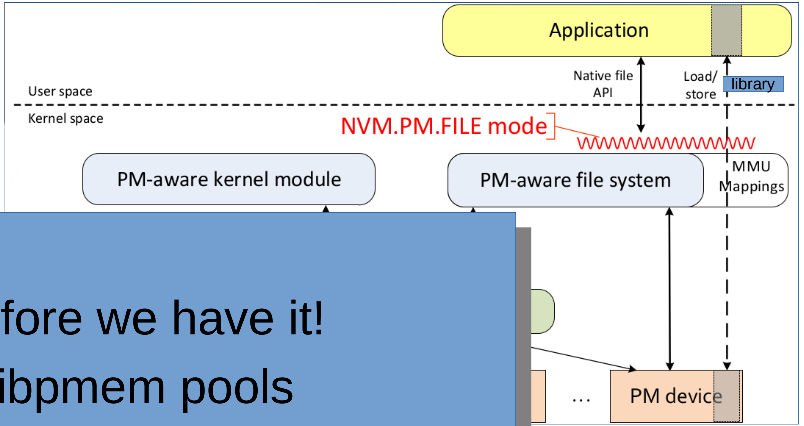
- NVM.PM.FILE
  - Simple solution for security & finding data in PMEM again
  - PM-aware filesystem allows direct access (no page-cache, mmaped region is byte addressable)
  - Linux DAX support for ext4, xfs, ...



- NVML library
  - libpmem: low level PMEM support, basis for all other libs
  - libpmemobj: transactional object store, providing memory allocation, transactions, and general facilities for PMEM programming
  - libpmemblk: arrays of same-size blocks, atomically updated, libpmemlog: PM-resident log file
  - libvmem: turns pool of PM into volatile memory pool, own malloc API
  - libvmmalloc: transparently converts all dynamic memory allocations into persystent memory
  - librpmem: remote access to remote persystent memory through RDMA

PMEM Programming Model

- NVM.PM.FILE
  - Simple solution for security & finding data in PMEM again
  - PM-aware filesystem allows direct



Libraries:

- We can develop for NVM/PMEM before we have it!
- Any file/filesystem can be used for libpmem pools (as long as mmap‘able)
- Performance penalty due to page-faults & page cache
- NVMe (& tradeoffs)… best we can get
- libvmem: first steps, if we don‘t care about persistence
- libpmemmobj + librpmem

n,

PI

; into persystent

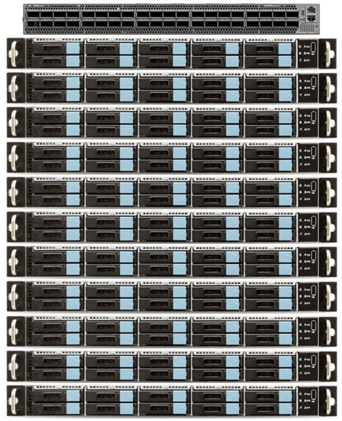
memory

- librpmem: remote access to remote persystent memory through RDMA

\Orchestrating a brighter world NEC

Conclusion

- We can build the post-processing cluster for ~100TB outputs!
  - Not yet with NVDIMMs with byte addressed access (1-2 years?)
  - Use NVMes for now: prices are in free fall (<1$/GB)
  - Rough price: ~150k€, in 12-14U
- Generic, multi-purpose building block:
  - 1U server, 2 Intel CPUs, 64 PCIe gen3 lanes,
    32 for 8 NVMes, 32 for 2 IB cards
- Is it fast enough?
  - Of course not. Bandwidth limited. And that is scalable.
  - NVMe over Fabrics not really helpful here. PMEM is the real deal.



\Orchestrating a brighter world

NEC